# Scientific Seminar "Bayesian Methods of ML"

# Deep Structured Models

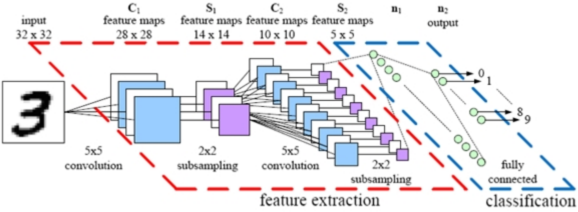
Ashuha Arseniy

Moscow Institute of Physics and Technology

ars.ashuha@gmail.com

April 28, 2016

Based on article: *Chen, Schwing, et al. "Learning deep structured models." ICML 2015*




- **NNs** is a framework for constructing flexible models
- Neural net is a **composition** linear and nonlinear functions

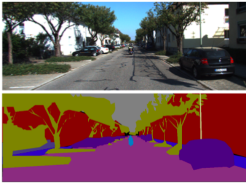
$argmax(\sigma[Linear(\sigma[Linear($  $, w)], w)]) = \text{cat}$

- **We can learn it efficiently** by back propagation

**Problem**

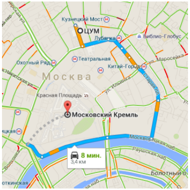
Can't take into account dependences between predicted variables.

- Non structured – predict simple variable (like a number)
- Structured – predict difficult variable (like a matrix, tree, sequence)

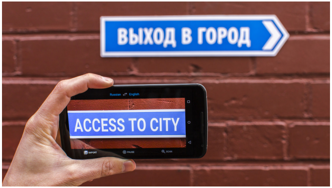


(a) Segmentation, $|Y| = \#pixel^{\#sigment}$


(b) Tagging, $|Y| = 2^{\#tags}$

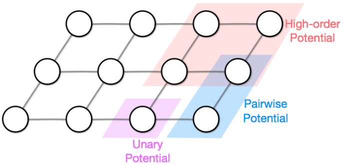



(c) Traffic prediction, $|Y| = ?$




(d) Translation, $|Y| = ?$

- **GMs** is a framework for taking into account dependencies between predicted variables.
- What can we do with exp-large output space? Use local dependences.
- Introduce prior knowledge as a score functions $\phi_r(y_r)$, $|y_r|$ is small




$\phi_{y1,y2}(people, male) = 10$ is high
$\phi_{y1,y2}(wather, girl) = 0.2$ is low
....

- We can introduce non-normalized probability distribution over outputs

$$p(y|x,w) = \frac{1}{Z}\prod_r \phi_r(x, y_r; w) \qquad Energy = -\sum_r \phi_r(x, y_r; w)$$

- **Inference Task**:

$$y^\star = \arg\max_y p(y|x,w)$$

1. We want to **train parameters** $w$ of parametric potential
2. Given training data $(x, y) \in D$; estimate the functions $f_r(y, x, w)$
3. Minimize a typically convex loss and a regularize on training set

$$Loss_{log}(x, y, w) = -\ln \; p_{x,y}(y; w)$$

$$Loss_{hinge}(x, y, w) = \max_{\hat{y}}(\Delta(y, \hat{y}) - w^T\Phi(x, \hat{y}) + w^T\Phi(x, y))$$

4. The assumption is that the model is log-linear

$$E(x, y, w) = -w^T\phi(x, y)$$

and the features decompose in a graph

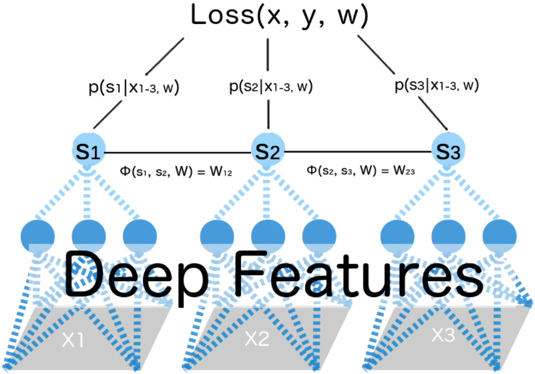
$$w^T\phi(x, y) = \sum_{r \in R} w_r^t \phi(x, y)$$

**Problem**

How can we remove the log-linear restriction,
to use potentials such as Neural Nets?

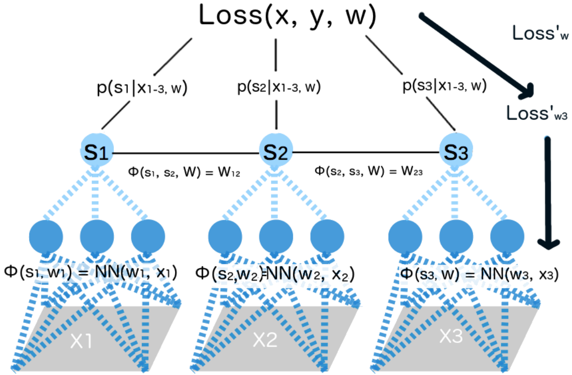
How can we combine Graphical Models and Deep Neural Nets?

1. Peace-wise learning:
   - train deep features $\rightarrow$ train linear potential $\rightarrow$ inference in GM
2. Jointly learning:
   - train deep features as non linear potential $\rightarrow$ inference in GM



How can we combine Graphical Models and Deep Neural Nets?

1. Peace-wise learning:
   - train deep features → train linear potential → inference in GM
2. Jointly learning:
   - train deep features as non linear potential → inference in GM



- We have: scoring function $F(x, y; w)$, training data $(x, y) \in D$
- Prediction proses is equal finding maximum scoring configuration $y^\star$:

$$y^\star = arg \max_y F(x, y; w)$$

- Introduce probability distribution over configurations as

$$p_{(x,y)}(\hat{y}|w) = \frac{exp\ F(x, \hat{y}, w)}{\sum_{y'} exp\ F(x, y', w)} = \frac{exp\ F(x, \hat{y}, w)}{Z(x, w)}$$

  rephrase previous task as finding high probably configuration
- Training proses is finding parameters $w$ by MLE

$$w = \arg\max_w log \prod_{(x,y)\in D} p_{(x,y)}(y|w) =$$
$$= \arg\max_w \sum_{(x,y)\in D} F(x, y, w) - ln\ Z(x, w)$$

We have optimization problem:

$$\sum_{(x,y)\in D} \left( F(x,y,w) - log \sum_{y'\in Y} exp\ F(x,y',w) \right) \to \max_w$$

Let's solve it by gradient assent (will be proof on the board if it's necessary):

$$\frac{\partial}{\partial w} \sum_{(x,y)\in D} (F(x,y,w) - logZ(x,w)) =$$

$$= \sum_{(x,y)\in D} \sum_{y'\in Y} (p(y'|w,x) - \delta(y'=y)) \frac{\partial}{\partial w} F(x,y',w)$$

**Very easy! Where is a challenge?**

Problem: What If Y is exponentially large!

1) How can we represent F? 2) What we can do with sum over Y?

$$\sum_{(x,y)\in D}\left(F(x,y,w) - log \sum_{y'\in Y} exp\ F(x,y',w)\right) \to \max_w$$

1. Use the graphical model $F(x,y;w) = \sum_r f_r(x,y;w)$

$$\frac{\partial}{\partial w}\sum_{(x,y)\in D}(F(x,y,w) - logZ(x,w)) =$$

$$= \sum_{(x,y)\in D}\sum_{y'\in Y}(p(y'|w,x) - \delta(y'=y))\frac{\partial}{\partial w}F(x,y',w)$$

(will be proof on the board if it's necessary):

$$= \sum_{(x,y)\in D}\sum_{y'_r,r}(p_r(y'_r|w,x) - \delta(y'_r=y_r))\frac{\partial}{\partial w}f_r(x,y'_r,w)$$

2. How to obtain marginals $p_r(y_r|w,x)$?
3. Use beliefs $p_r(y_r|w,x) \approx b_r(y_r|w,x)$

## Deep Structured Learning (algo 1)

Repeat until stopping criteria:

1. Forward pass to compute the $f_r(y_r, x; w)$ $\quad \forall r, y_r, (x, y) \in D$
2. Compute the $b_r(y_r|x, w)$ by approx inference $\forall r, y_r, (x, y) \in D$
3. Backward pass via chain rule to obtain gradient

$$\frac{\partial}{\partial w} = \sum_{(x,y)\in D, y'_r, r} (b_r(y'_r|w, x) - \delta(y'_r = y_r)) \frac{\partial}{\partial w} f_r(x, y'_r; w)$$

4. Update parameters w

$$w = w - \alpha \cdot \partial / \partial w$$

## Problem

We run inference for each object to make one parameters update

$$\sum_{(x,y)\in D}\left(F(x,y,w) - log \sum_{y'\in Y} exp\ F(x,y',w)\right) \to \max_w$$

1. We can represent Z as (will be proof on the board if it's necessary):

$$ln\ Z = \sum_{\hat{y}} exp\ F(x,\hat{y},w) = \max_{p_{(x,y)}} \mathbb{E}_{p_{(x,y)}(\hat{y})} F(x,\hat{y};w) + H(p_{(x,y)})$$

2. Assumption, F and H is decomposed into a sum of "local" functions

$$F = F(x,y;w) = \sum_r f_r(x,y_r;w) \qquad H = H(p_{(x,y)}) = \sum_r H(p_{(x,y),r})$$

3. Rephrase our task as

$$\min_w \sum_{(x,y)\in D}\left(\max_{p_{(x,y)}}\left\{\sum_r p_{(x,y),r}(\hat{y}_r) f_r(x,\hat{y}_r;w) + H(p_{(x,y)})\right\} - F\right)$$

$$\sum_{(x,y)\in D}\left(\max_{p_{(x,y)}}\left\{\sum_r p_{(x,y),r}(\hat{y}_r)f_r(x,\hat{y}_r;w)+H(p_{(x,y)})\right\}-F\right)\to\min_w$$

1. We can't compute true marginals, let's use beliefs $b_{(x,y)} \approx p_{(x,y)}$

$$b_{(x,y)} \in C_{(x,y)} = \begin{cases} b_{(x,y),r}(\cdot) \geq 0 \quad \sum_{y_r} b_{(x,y),r}(y_r) = 1 & \forall r \\ b_{(x,y),r} = \sum_{\hat{y}_p \setminus \hat{y}_r} p_{(x,y),p}(\hat{y}_p) & \forall r, \hat{y}_r, p \in P(r) \end{cases}$$

2. $P(r) = \{p \in Y : r \subset p\}$ and $C(r) = \{c \in Y : r \in P(c)\}$
3. Rephrase our task as

$$\min_w \sum_{(x,y)\in D}\left(\max_{b_{(x,y)}\in C_{(x,y)}}\left\{\sum_r b_{(x,y),r}(\hat{y}_r)f_r(x,\hat{y}_r;w)+H(b_{(x,y)})\right\}-F\right)$$

**Problem**

We need to solve inner problem to compute subgradient!

$$\min_{w} \sum_{(x,y)\in D} \left( \max_{b_{(x,y)}\in C_{(x,y)}} \left\{ \sum_{r} b_{(x,y),r}(y_r) f_r(x, y_r; w) + H(b_{(x,y)}) \right\} - F \right)$$

s.t. $b_{(x,y)} \in C_{(x,y)}$ marginalization and discrete distribution conditions

H is redefined as barrier function when argument is not a distribution:

1. The Lagrangian of inner problem is:

$$L_{(x,y)} = \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \cdot \hat{f}_r(x, \hat{y}_r; w, \lambda) + H_{barier}$$

$$\hat{f}_r(x, \hat{y}_r; w, \lambda) = f_r(x, \hat{y}_r; w) + \sum_{p\in P(r)} \lambda_{(x,y),p\to r}(\hat{y}_r) - \sum_{c\in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c)$$

2. Move to dual task by $\lambda$ ($ln\ Z = \max_{p_{(x,y)}} \mathbb{E}_{p_{(x,y)}(\hat{y})} F(x, \hat{y}; w) + H(p_{(x,y)})$):

$$\min_{w,\lambda} \sum_{(x,y),r} \ln \sum_{\hat{y}_r} exp\ \hat{f}_r(x, \hat{y}_r; w, \lambda) - \sum_{(x,y)\in D} F(x, y; w)$$

Standard learning:

repeat
- repeat
  - update marginals $b_r$
- until convergence
- update parameters w

until convergence

Blended learning:

repeat
- update marginals $b_r$
- update parameters w

until convergence

**Advantage**:

More frequent parameter updates

---

Hazan, Schwing, McAllester, Urtasun: Blending Learning and Inference in Structured Prediction

$$D(\lambda, w) = \sum_{(x,y),r} \ln \sum_{\hat{y}_r} exp(f_r(x, \hat{y}_r; w) + \sum_{c \in C(r)} \lambda_{(x,y),c \to r}(\hat{y}_c) - \sum_{p \in P(r)} \lambda_{(x,y),r \to p}(\hat{y}_r)) - \sum_{(x,y)} F(x, y; w) \to \min_{w,\lambda}$$

- Optimize by $w$ (will be proof on the board if it's necessary):

$$\frac{\partial D}{\partial w} = \sum_{(x,y),r,\hat{y}_r} b_{(x,y),r,\hat{y}_r} \frac{\partial}{\partial w} f_r(x, \hat{y}_r; w) + \sum_{(x,y)} \frac{\partial}{\partial w} F(x, y; w)$$

- Optimize by $\lambda$ (will be proof on the board if it's necessary):

$$\mu_{(x,y),p \to r}(\hat{y}_r) = \ln \sum_{\hat{y}_p \setminus \hat{y}_r} \exp\left(f_p(x, \hat{y}_p; w) - \sum_{p' \in P(p)} \lambda_{(x,y),p \to p'}(\hat{y}_p) + \sum_{r' \in C(p) \setminus r} \lambda_{(x,y),r' \to p}(\hat{y}_{r'})\right)$$

$$\lambda_{(x,y),r \to p}(\hat{y}_r) \propto c_r \cdot (f_r(x, \hat{y}_r; w) - \sum_{c \in C(r)} \lambda_{(x,y),c \to r}(\hat{y}_c) + \sum_{p \in P(r)} \mu_{(x,y),p \to r}(\hat{y}_r)) - \mu_{(x,y),p \to r}(\hat{y}_r)$$

## Efficient Deep Structured Learning (algo 2)

Repeat until stopping criteria:

1. Forward pass to compute the $f_r(y_r, x; w)$ $\quad \forall r, y_r, (x, y) \in D$
2. Compute the $b_r(y_r|x, w) = \exp(\hat{f}_r(x, y_r; w, \lambda))$ $\forall r, y_r, (x, y) \in D, p \in P(r)$

$$\mu_{(x,y),p\to r}(\hat{y}_r) = \ln \sum_{\hat{y}_p \setminus \hat{y}_r} \exp\left(f_p(x, \hat{y}_p; w) - \sum_{p' \in P(p)} \lambda_{(x,y),p\to p'}(\hat{y}_p) + \sum_{r' \in C(p)\setminus r} \lambda_{(x,y),r'\to p}(\hat{y}_{r'})\right)$$

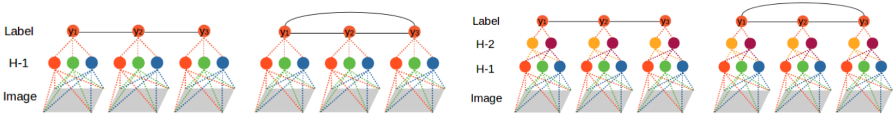
$$\lambda_{(x,y),r\to p}(\hat{y}_r) \propto c_r \cdot \left(f_r(x, \hat{y}_r; w) - \sum_{c\in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c) + \sum_{p\in P(r)} \mu_{(x,y),p\to r}(\hat{y}_r)\right) - \mu_{(x,y),p\to r}(\hat{y}_r)$$

3. Backward pass via chain rule to obtain gradient

$$g = \sum_{(x,y),r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \nabla_w f_r(\hat{y}_r, x; w) - \nabla_w \sum_{(x,y),r} f_r(x, y; w)$$

4. Update parameters w

$$w = w - \alpha \cdot \partial / \partial w$$




1. Modeling of correlations between variables
2. Non-linear dependence on parameters
3. Joint training of many convolution neural networks

- **Task**: Find a combination of tags that describe the image, $|Y| = 2^{38}$

female/indoor/portrait
female/indoor/portrait

sky/plant life/tree
sky/plant life/tree

water/animals/sea
water/animals/sky

- **Graphical Model**: Fully Connected 38
- **First order potential**: $f_i(x, y_i; U) = Alexnet(x, U)$
- **Second order potential**: $f_{i,j}(x, y_i, y_j; W) = W_{y_i y_j}$

| Training method | Prediction error [%] |
|---|---|
| Unary only | 9.36 |
| Piecewise | 7.70 |
| Joint (with pre-training) | **7.25** |

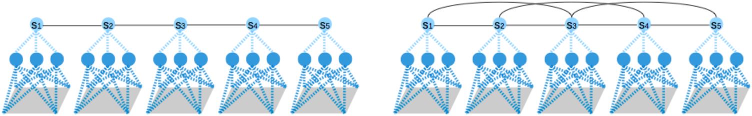
Learned class "correlations":

| | female | people | indoor | portrait | sky | plant life | male | clouds | tree |
|---|---|---|---|---|---|---|---|---|---|
| female | 0.00 | 0.68 | 0.04 | 0.24 | -0.01 | -0.05 | 0.07 | -0.01 | 0.01 |
| people | 0.68 | 0.00 | 0.06 | 0.36 | -0.05 | -0.12 | 0.74 | -0.04 | -0.03 |
| indoor | 0.04 | 0.06 | 0.00 | 0.07 | -0.35 | -0.34 | 0.02 | -0.15 | -0.21 |
| portrait | 0.24 | 0.36 | 0.07 | 0.00 | -0.02 | -0.01 | 0.12 | 0.02 | 0.05 |
| sky | -0.01 | -0.05 | -0.35 | -0.02 | 0.00 | 0.24 | -0.00 | 0.44 | 0.30 |
| lant life | -0.05 | -0.12 | -0.34 | -0.01 | 0.24 | 0.00 | -0.07 | 0.09 | 0.68 |
| male | 0.07 | 0.74 | 0.02 | 0.12 | -0.00 | -0.07 | 0.00 | 0.00 | -0.02 |
| clouds | -0.01 | -0.04 | -0.15 | 0.02 | 0.44 | 0.09 | 0.00 | 0.00 | 0.11 |
| tree | 0.01 | -0.03 | -0.21 | 0.05 | 0.30 | 0.68 | -0.02 | 0.11 | 0.00 |

- **Task**: Find five letters within distorted images, $|Y| = 26^5$

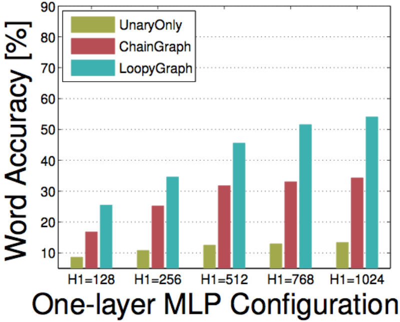
banal  julep  resty

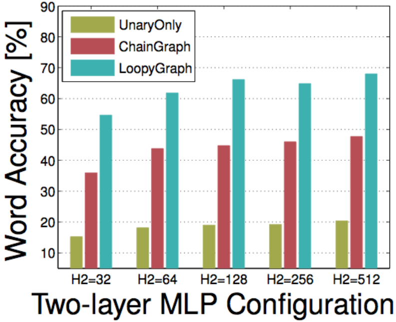
- **Graphical Model**:



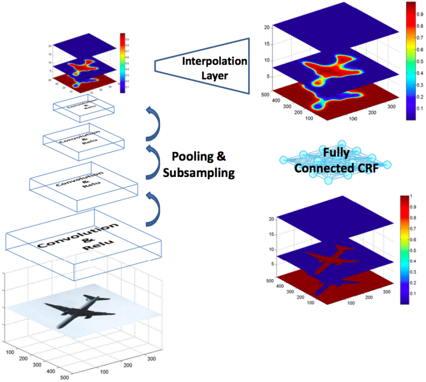

1st order Markov  2nd order Markov

- **First order potential**:
  1. One Layer : $f_i(x, y_i; U) = ReLu(U_1^T \cdot x)$
  2. Two Layers: $f_i(x, y_i; U) = ReLu(U_2^T \cdot ReLu(U_1^T \cdot x))$
- **Second order potential**:
  1. Linear: $f_{i,j}(x, y_i, y_j; W) = W_{y_i y_j}$


Word Accuracy [%]
UnaryOnly
ChainGraph
LoopyGraph
H1=128
H1=256
H1=512
H1=768
H1=1024
One-layer MLP Configuration


Word Accuracy [%]
UnaryOnly
ChainGraph
LoopyGraph
H2=32
H2=64
H2=128
H2=256
H2=512
Two-layer MLP Configuration

- **Task**: Image segmentation
- **Graphical Model**: Fully connected CRF with Gaussian potentials
- **NN**: PreTrain OxfordNet , predicts $40 \times 40$ + upsampling
- **Inference**: using (algo1), with mean-field as approx. inference




| Training method | Mean IoU [%] |
|---|---|
| Unary only | 61.476 |
| Joint | **64.060** |

female/indoor/portrait female/indoor/portrait | sky/plant life/tree sky/plant life/tree | water/animals/sea water/animals/sky

1. Jointly learning helps
2. Non-linear pairwise function improves over the linear one
3. Deeper and more structured $\rightarrow$ better performance
4. Wide range of applications: Word recognition, Tagging, Segmentation

- Chen, Schwing, Learning Deep Structured Models v1 v2 v3 icml
- Hazan, Schwing, Blending Learning and Infer. in Struct. Pred. paper
- Raquel Urtasun, CS Department, UofT, Learning Deep SM slides
- Liang-Chieh Chen, CS Department, UofC, ICML video slides
- Alexandr Schwing, CS Department, UofT, Re.Work video